Data Analysis and Visualization Exercise 11&12

Daniela Klaproth-Andrade, Felix Brechtmann, Julien Gagneur

Section 00 - Getting ready

1. Make sure you have already installed and loaded the following libraries:

```
library(ggplot2)
library(data.table)
library(magrittr)
library(tidyr)
library(ggrepel)

library(caret)
library(plotROC)

library(randomForest)
library(rpart)
```

Section 01 - Logistic regression on Diabetes dataset

In this section we are considering the dataset `pima-indians-diabetes.csv` which is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. A more detailed description of the data can be obtained from Kaggle: https://www.kaggle.com/uciml/pima-indians-diabetes-database.

Load the dataset with the following lines of code:

```
diabetes_dt <- fread("extdata/pima-indians-diabetes.csv")
diabetes_dt[, Outcome := as.factor(Outcome)]

# Store feature variables that we will need for later
feature_vars <- colnames(diabetes_dt[,-c("Outcome")])

diabetes_dt
```

```
##      Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
##   1:           6     148            72            35       0 33.6
##   2:           1      85            66            29       0 26.6
##   3:           8     183            64             0       0 23.3
##   4:           1      89            66            23      94 28.1
##   5:           0     137            40            35     168 43.1
##  ---
## 764:          10     101            76            48     180 32.9
## 765:           2     122            70            27       0 36.8
## 766:           5     121            72            23     112 26.2
## 767:           1     126            60             0       0 30.1
## 768:           1      93            70            31       0 30.4
```

```
##      DiabetesPedigreeFunction Age Outcome
##   1:                    0.627  50       1
##   2:                    0.351  31       0
##   3:                    0.672  32       1
##   4:                    0.167  21       0
##   5:                    2.288  33       1
##  ---
## 764:                    0.171  63       0
## 765:                    0.340  27       0
## 766:                    0.245  30       0
## 767:                    0.349  47       1
## 768:                    0.315  23       0
```

1. Is the diabetes dataset balanced?

```
diabetes_dt[, .N, by=Outcome] # absolute numbers for each class
```

```
##    Outcome   N
## 1:       1 268
## 2:       0 500
```

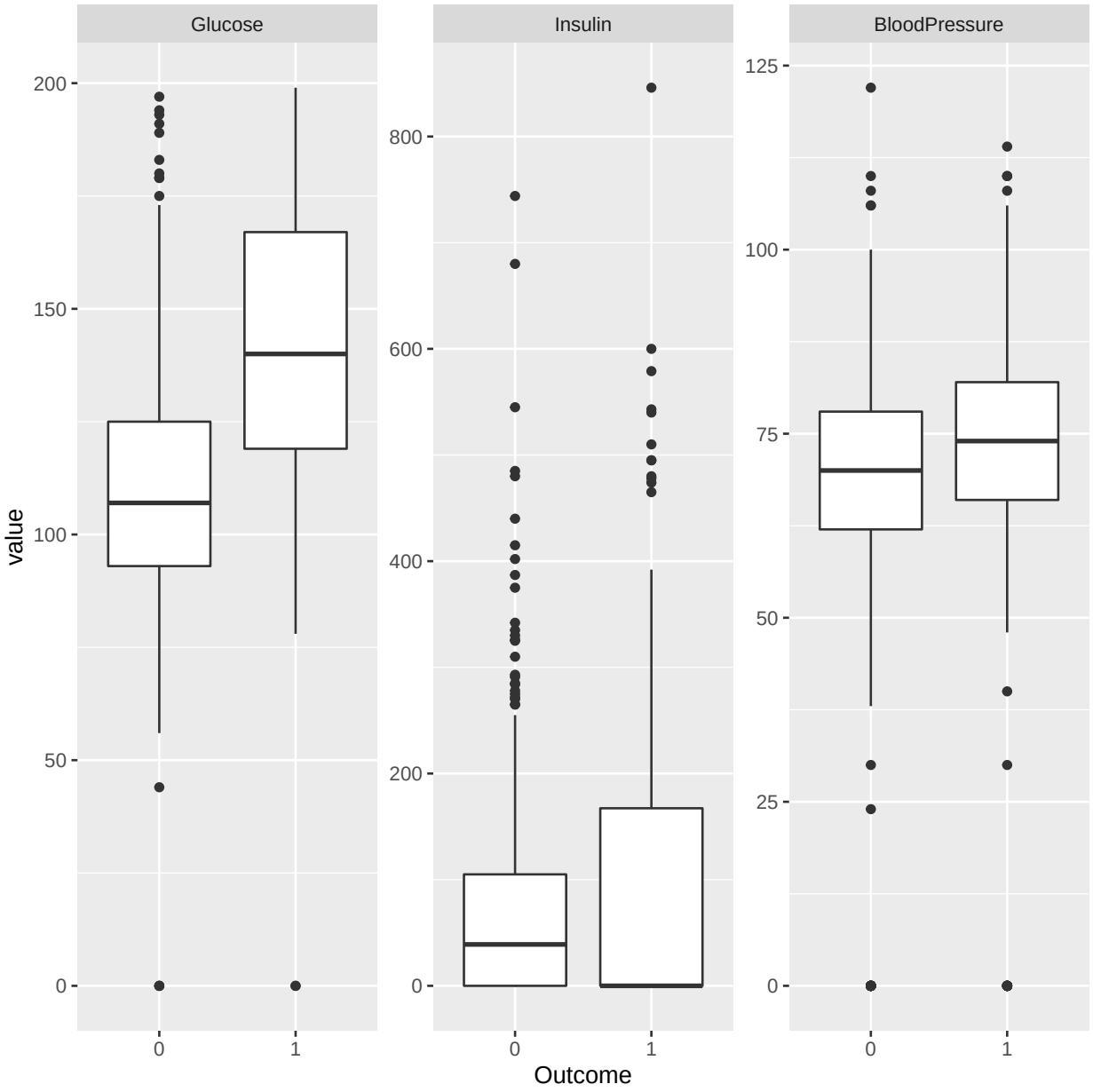
```
diabetes_dt[, .N/nrow(diabetes_dt), by=Outcome] # class proportions
```

```
##    Outcome        V1
## 1:       1 0.3489583
## 2:       0 0.6510417
```

```
# No, the dataset is imbalanced
```

2. Create an appropriate plot to visualize the relationship between the `Outcome` variable and the feature variables `Glucose`, `BloodPressure` and `Insulin`. What do you conclude from your visualization?

```
## Boxplot/violin since we have a binary outcome variable and numeric feature variables
melted_diabetes_dt <- melt(diabetes_dt[, .(Glucose,
                                           Insulin, Outcome, BloodPressure)],
                           id.vars="Outcome")
ggplot(melted_diabetes_dt, aes(Outcome, value)) + geom_boxplot() +
  facet_wrap(~variable, scales="free")
```




```
## Glucose seem to have a higher impact on the outcome variable.
```

3. Fit a logistic regression model for predicting `Outcome` only based on the feature `Glucose`. Inspect the coefficients of the model's predictors. What do these coefficients mean?

```
# Fit logistic regression models for GLucose
logreg_1 <-  glm(Outcome~Glucose,
                 data = diabetes_dt, family = "binomial")

# Have a first look at the output of the model
logreg_1
```

```
##
## Call:  glm(formula = Outcome ~ Glucose, family = "binomial", data = diabetes_dt)
##
```

```
## Coefficients:
## (Intercept)      Glucose
##    -5.35008      0.03787
##
## Degrees of Freedom: 767 Total (i.e. Null);  766 Residual
## Null Deviance:       993.5
## Residual Deviance: 808.7     AIC: 812.7
```

```
summary(logreg_1)
```

```
##
## Call:
## glm(formula = Outcome ~ Glucose, family = "binomial", data = diabetes_dt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1096  -0.7837  -0.5365   0.8566   3.2726
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.350080   0.420827  -12.71   <2e-16 ***
## Glucose      0.037873   0.003252   11.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 808.72  on 766  degrees of freedom
## AIC: 812.72
##
## Number of Fisher Scoring iterations: 4
```

```
# Inspect coefficients
coeffs <- logreg_1$coefficients
coeffs
```

```
## (Intercept)     Glucose
## -5.35008039  0.03787304
```

```
# Translate the coefficients to odds ratios
odd_ratios <- exp(coeffs)
odd_ratios
```

```
## (Intercept)     Glucose
## 0.004747769 1.038599360
```

```
# In this case we have one intercept value and one coeff for each feature
# i.e. only one coeff for this simple model.
# The model coefficients can be interpreted as log odds ratios,
# which can be easily transformed to odds ratios be exponentiating them.
# Those odds ratios tell you by what ratio the odds of the outcomes
# change when changing the predictive variable by one unit.
# Scale each feature by sd to compare feature odd ratios
```

4. Create two further logistic regression models for predicting `Outcome`. For one model, use only the feature variable `BloodPressure` for building the model. For the other model, use only the feature variable `Insulin`.

Which models have a significant feature?

```
# Fit two further models with different features
logreg_2 <-  glm(Outcome~BloodPressure,
                 data = diabetes_dt, family = "binomial")
logreg_3 <-  glm(Outcome~Insulin,
                 data = diabetes_dt, family = "binomial")

logreg_2
```

```
##
## Call:  glm(formula = Outcome ~ BloodPressure, family = "binomial", data = diabetes_dt)
##
## Coefficients:
##   (Intercept)  BloodPressure
##     -1.140092       0.007425
##
## Degrees of Freedom: 767 Total (i.e. Null);  766 Residual
## Null Deviance:       993.5
## Residual Deviance: 990.1     AIC: 994.1
```

```
summary(logreg_2)
```

```
##
## Call:
## glm(formula = Outcome ~ BloodPressure, family = "binomial", data = diabetes_dt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0797  -0.9389  -0.9000   1.4097   1.6838
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.140092   0.299822  -3.803 0.000143 ***
## BloodPressure  0.007425   0.004141   1.793 0.072994 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 990.13  on 766  degrees of freedom
## AIC: 994.13
##
## Number of Fisher Scoring iterations: 4
```

```
logreg_3
```

```
##
## Call:  glm(formula = Outcome ~ Insulin, family = "binomial", data = diabetes_dt)
##
## Coefficients:
## (Intercept)      Insulin
##   -0.814510     0.002299
##
## Degrees of Freedom: 767 Total (i.e. Null);  766 Residual
```

```
## Null Deviance:       993.5
## Residual Deviance: 980.8     AIC: 984.8
```

```
summary(logreg_3)
```

```
##
## Call:
## glm(formula = Outcome ~ Insulin, family = "binomial", data = diabetes_dt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5736  -0.9129  -0.8563   1.3761   1.5370
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.8145101  0.0943584  -8.632  < 2e-16 ***
## Insulin      0.0022988  0.0006535   3.518 0.000435 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 980.81  on 766  degrees of freedom
## AIC: 984.81
##
## Number of Fisher Scoring iterations: 4
```

```
# Glucose and Insulin are significant. Blood preassure is not.
```

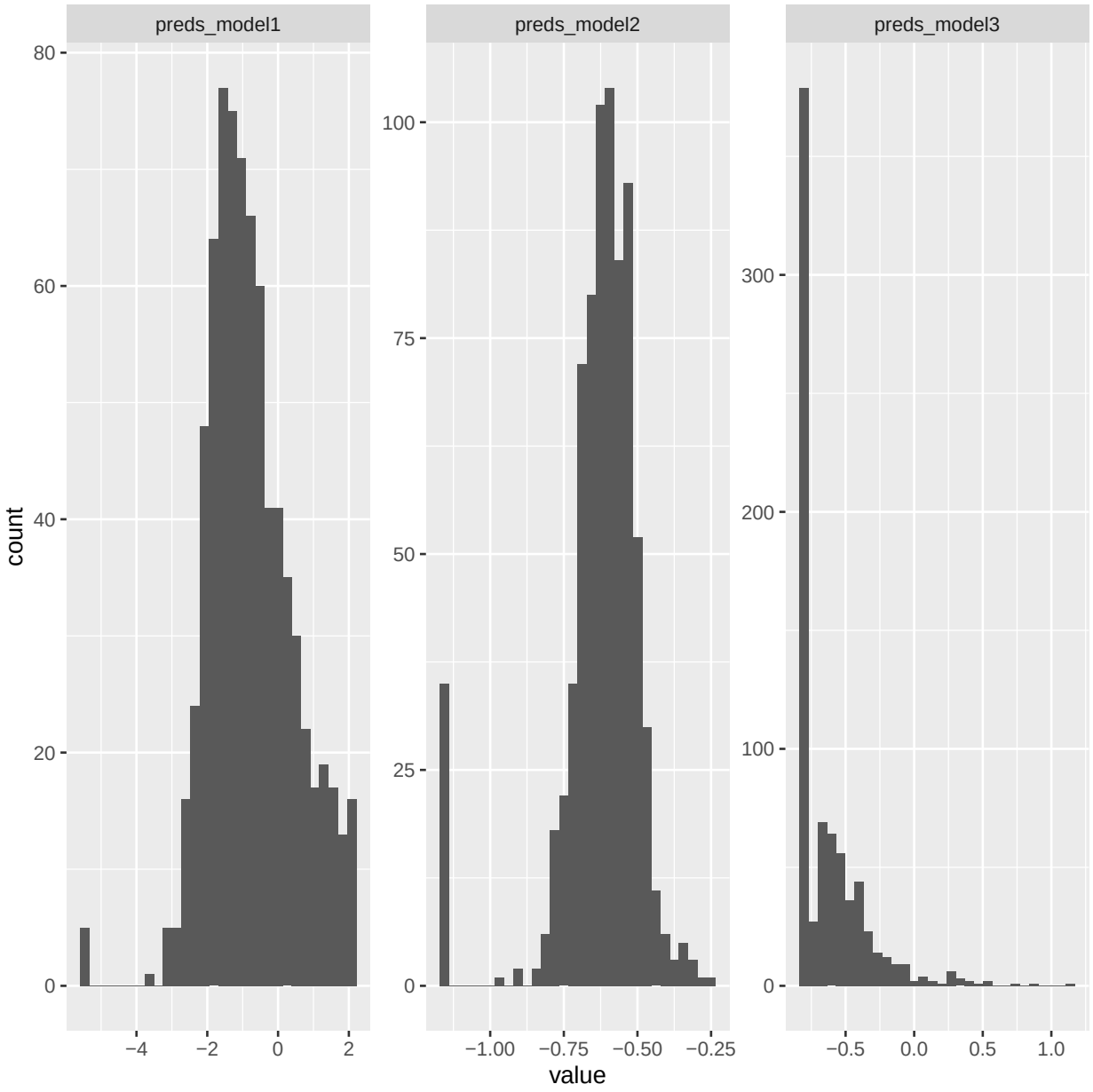
5. Collect the predictions of each model for all samples in the dataset. Store the scores of each model in a separate column of the original dataset. Visualize the distributions of the scores with an appropriate plot. Which type of distribution would you ideally expect? Hint: Use the `predict()` function.

```
diabetes_dt[, preds_model1 := predict(logreg_1)]
diabetes_dt[, preds_model2 := predict(logreg_2)]
diabetes_dt[, preds_model3 := predict(logreg_3)]
diabetes_dt
```

```
##      Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
##   1:           6     148            72            35       0 33.6
##   2:           1      85            66            29       0 26.6
##   3:           8     183            64             0       0 23.3
##   4:           1      89            66            23      94 28.1
##   5:           0     137            40            35     168 43.1
##  ---
## 764:          10     101            76            48     180 32.9
## 765:           2     122            70            27       0 36.8
## 766:           5     121            72            23     112 26.2
## 767:           1     126            60             0       0 30.1
## 768:           1      93            70            31       0 30.4
##      DiabetesPedigreeFunction Age Outcome preds_model1 preds_model2
##   1:                    0.627  50       1    0.2551290   -0.6055098
##   2:                    0.351  31       0   -2.1308723   -0.6500583
##   3:                    0.672  32       1    1.5806852   -0.6649078
##   4:                    0.167  21       0   -1.9793802   -0.6500583
```

```
##   5:                    2.288  33       1   -0.1614744   -0.8431018
##  ---
## 764:                    0.171  63       0   -1.5249037   -0.5758108
## 765:                    0.340  27       0   -0.7295700   -0.6203593
## 766:                    0.245  30       0   -0.7674430   -0.6055098
## 767:                    0.349  47       1   -0.5780778   -0.6946068
## 768:                    0.315  23       0   -1.8278880   -0.6203593
##      preds_model3
##   1:   -0.8145101
##   2:   -0.8145101
##   3:   -0.8145101
##   4:   -0.5984182
##   5:   -0.4283033
##  ---
## 764:   -0.4007171
## 765:   -0.8145101
## 766:   -0.5570389
## 767:   -0.8145101
## 768:   -0.8145101
```

```
# visualize predictions from different models.
ggplot(melt(diabetes_dt[, .(preds_model1, preds_model2, preds_model3)]), aes(value)) +
  geom_histogram() + facet_wrap(~variable, scales="free")
```




```
# Ideally, we would expect a bimodal distribution for separating
# classes from negative and positive models
```

6. Now, create a function for computing the confusion matrix based on the predicted scores of a model and the actual outcome. The function takes as input a threshold, a data table, the name of a scores column and the name of column with the actual labels. Then, use the implemented function for computing the confusion matrix of the first model for the thresholds -1, 0 and 1. Are there any differences? What is the amount of false positives for the last cutoff? You can use the following definition of the function:

```
confusion_matrix <- function(dt, score_column, labels_column, threshold){ }
```

```
confusion_matrix <- function(dt, score_column, labels_column, threshold){
  # The table() function is very useful for computing the confusion matrix
  # We have to use get() to get the column from a string
  return(dt[, table(get(labels_column), get(score_column)>threshold) ])
```

```
}

thresholds <- c(-1,0,1)
lapply(thresholds, function(t){confusion_matrix(diabetes_dt, "preds_model1", "Outcome", t)})
```

```
## [[1]]
##
##     FALSE TRUE
##   0   308  192
##   1    55  213
##
## [[2]]
##
##     FALSE TRUE
##   0   443   57
##   1   138  130
##
## [[3]]
##
##     FALSE TRUE
##   0   489   11
##   1   203   65
```

```
## For the last cutoff we obtain the following number of FP
confusion_matrix(diabetes_dt, "preds_model1", "Outcome", 1)["0", "TRUE"]
```

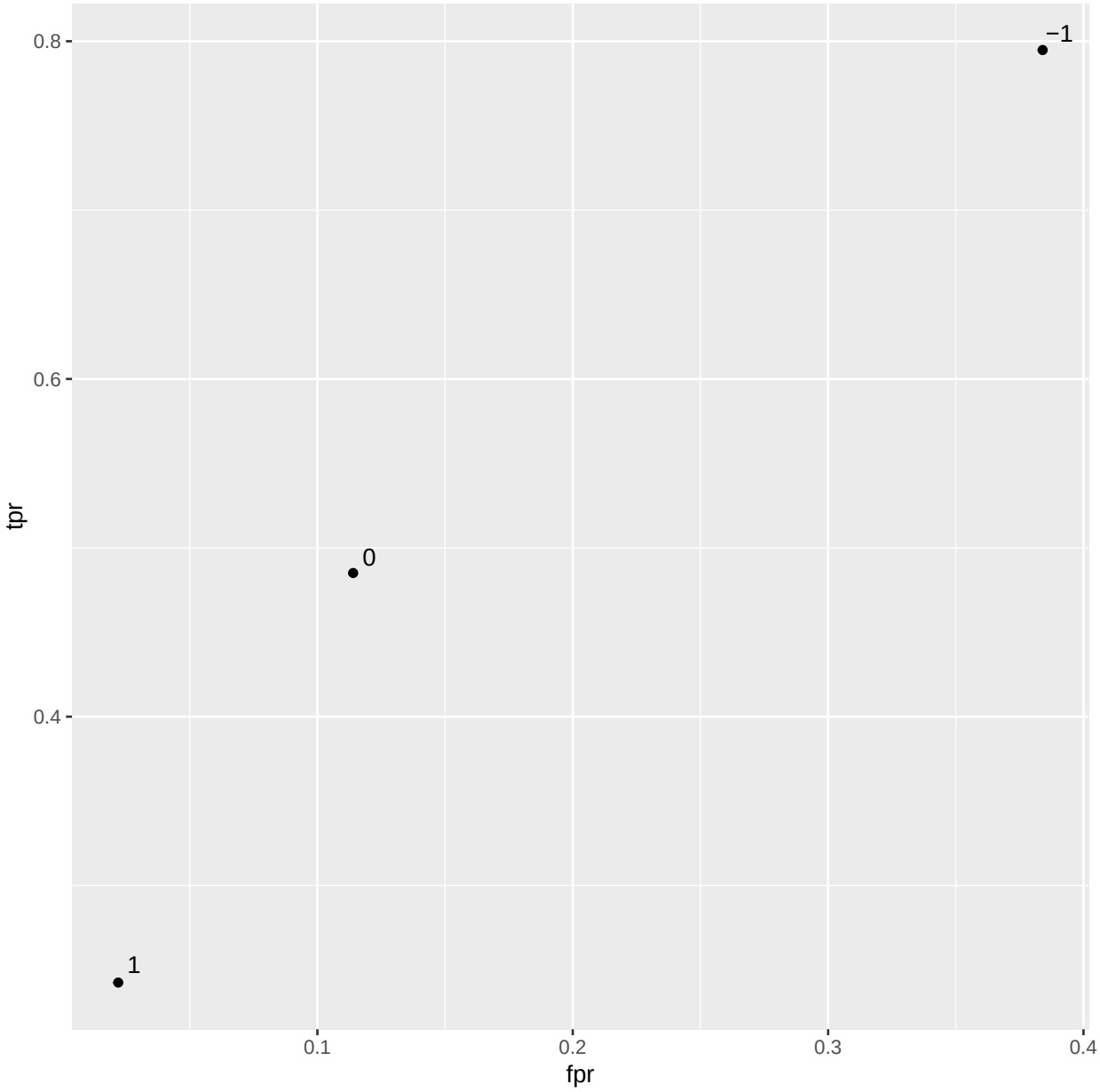
```
## [1] 11
```

7. Use the implemented function to create a second function for this time computing the TPR and FPR for a certain threshold of a classification model given the predicted scores of a model and the actual outcome. What is the TPR and the FPR of the first model for the thresholds -1, 0 and 1? Plot these values in a scatter plot. Your function should take the same parameters as before and return a data table as follows:

```
tpr_fpr <- function(dt, score_column, labels_column, threshold){
  tpr <- NULL # TODO
  fpr <- NULL # TODO
  return(data.table(tpr=tpr, fpr=fpr, t=threshold))
}
```

```
tpr_fpr <- function(dt, score_column, labels_column, threshold){
  # Use confusion matrix
  cm <- confusion_matrix(diabetes_dt, score_column, labels_column, threshold)

  # determine FP, TP, FN and TN from confusion matrix
  TP <- cm["1", "TRUE"]
  FP <- cm["0", "TRUE"]
  TN <- cm["0", "FALSE"]
  FN <- cm["1", "FALSE"]

  # compute FPR and TPR
  tpr <-  TP/(TP+FN)
  fpr <-  FP/(FP+TN)

  return(data.table(tpr=tpr, fpr=fpr, t=threshold))
}
```

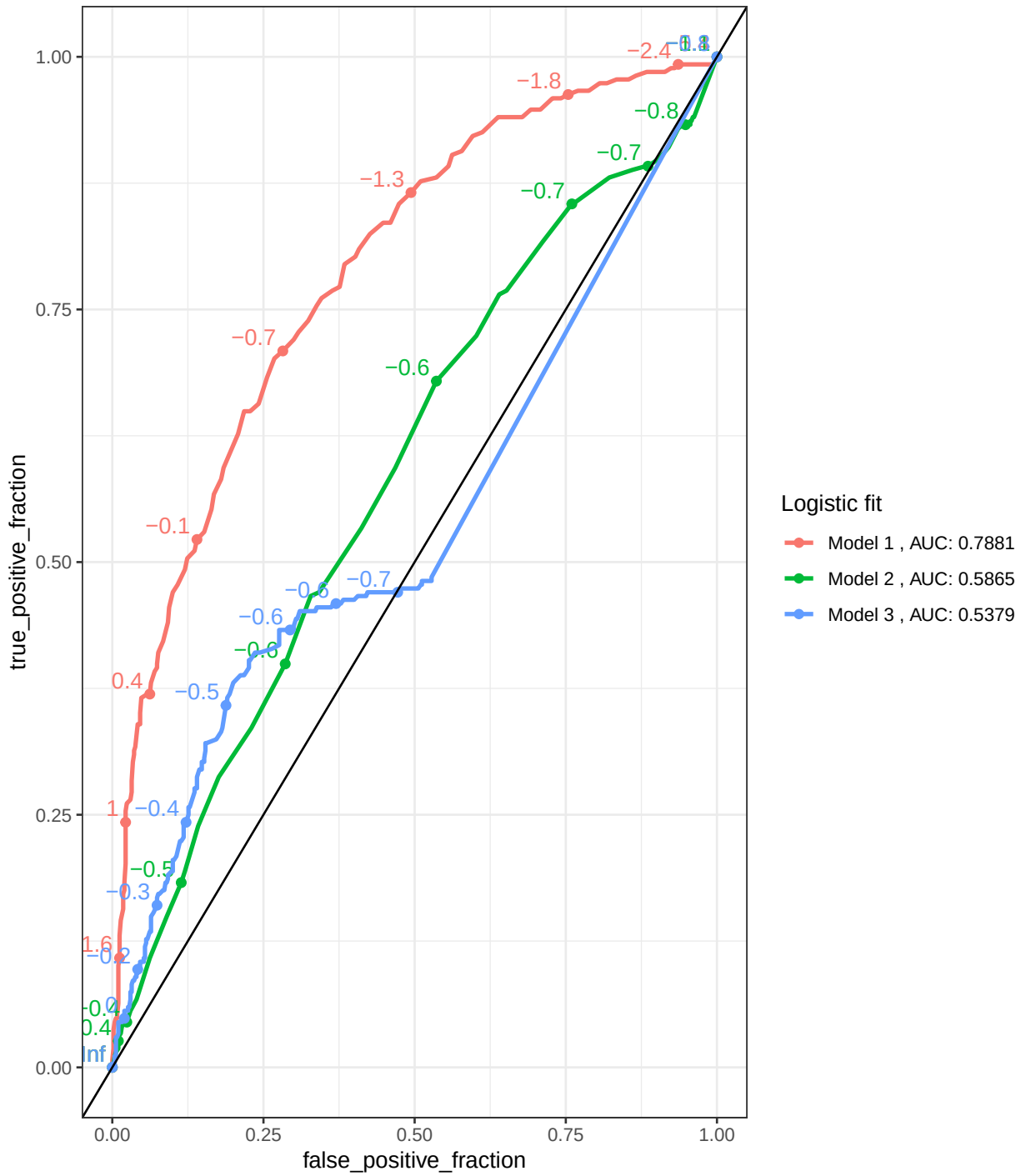
```
thresholds <- c(-1,0,1)
dt <- rbindlist(lapply(thresholds, function(t){ tpr_fpr(diabetes_dt, "preds_model1", "Outcome", t) }))
ggplot(dt, aes(fpr, tpr, label=t)) + geom_point() + geom_text_repel()
```



8. For a systematic comparison of the previously built three models, plot a ROC curve for each model into a single plot using the function `geom_roc` from the library `plotROC`. Add the area under the curve (AUC) to the plot. Which is the best model according to the AUC?

```
# Melt subset of diabetes_dt for plotting roc curves
plot_dt <- diabetes_dt[,.(Outcome, preds_model1, preds_model2, preds_model3 )] %>%
             melt(id.vars="Outcome", variable.name="logistic_fit", value.name="response")

# Plot roc curves for each model
ggroc <- ggplot(plot_dt, aes(d=as.numeric(Outcome), m=response, color=logistic_fit)) +
```

```
            geom_roc() +
            geom_abline() + theme_bw()
# geom_roc does not work with factors so we have to convert Outcome to numeric

# Add AUCs computed from function cal_auc
aucs <- as.data.table(calc_auc(ggroc))
# Add nice labels including AUC
labels <- sapply(1:3, function(i) {paste("Model", i, ", AUC:", round(aucs[group==i, AUC], 4)) } )
ggroc + scale_color_discrete(name = "Logistic fit", labels = labels)
```




```
# First model has the highest AUC
```

9. Now, fit a logistic regression model with all feature variables (stored in `feature_vars`). Visualize the distribution of the predicted scores for positive and negative classes. What can you conclude from this visualization regarding the separation of the two classes by the model? Plot once again the previous ROC curves and include the ROC curve of the full model for comparison.

```
full_formula <- as.formula(paste(c("Outcome ~ ",
                                   paste(feature_vars, collapse = " + ")),
```

```
                                  collapse = ""))
logreg_full <- glm(full_formula,
                   data = diabetes_dt, family = "binomial")

logreg_full
```

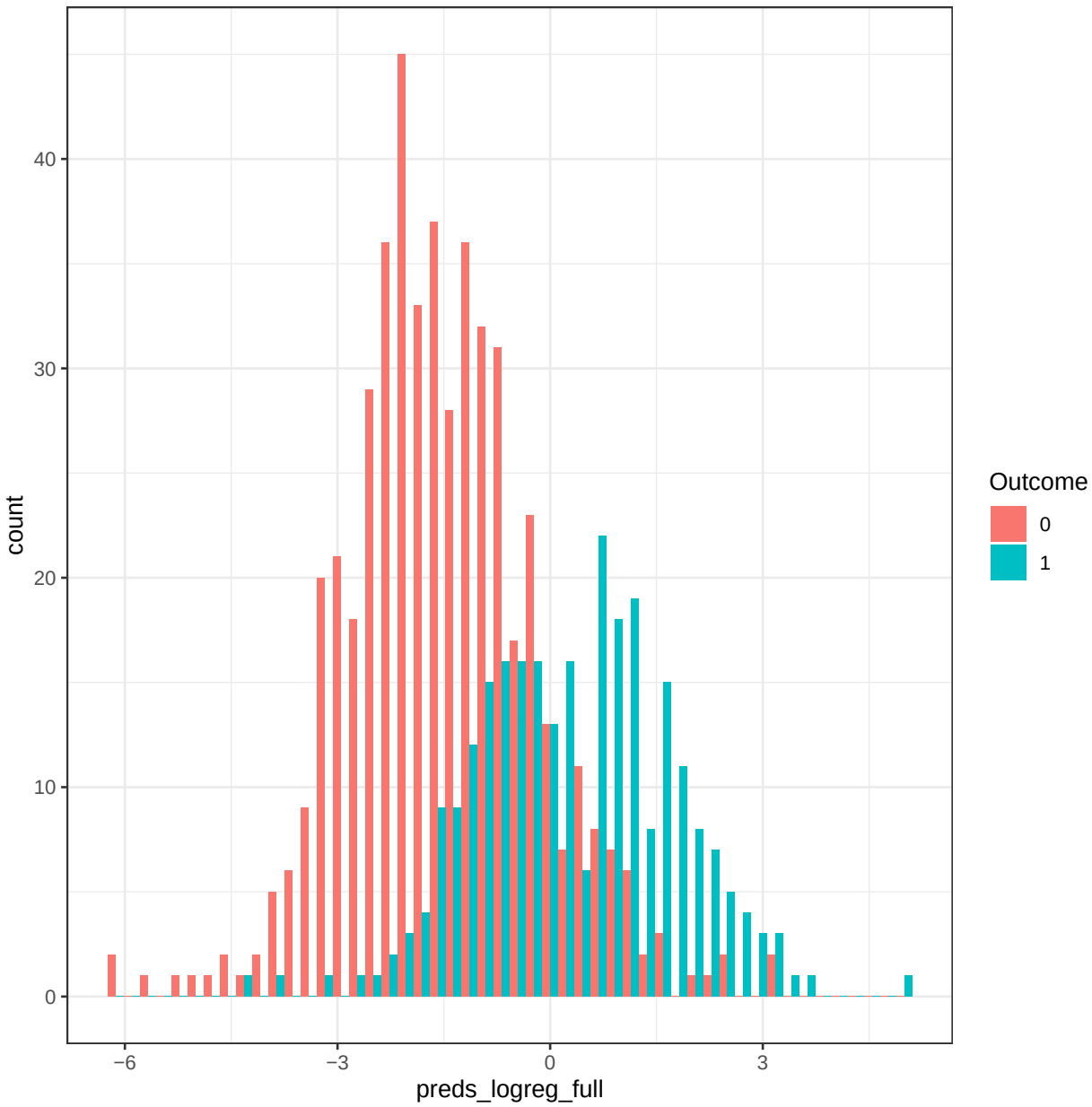
```
##
## Call:  glm(formula = full_formula, family = "binomial", data = diabetes_dt)
##
## Coefficients:
##               (Intercept)               Pregnancies                   Glucose
##                 -8.404696                  0.123182                  0.035164
##             BloodPressure             SkinThickness                   Insulin
##                 -0.013296                  0.000619                 -0.001192
##                       BMI  DiabetesPedigreeFunction                       Age
##                  0.089701                  0.945180                  0.014869
##
## Degrees of Freedom: 767 Total (i.e. Null);  759 Residual
## Null Deviance:       993.5
## Residual Deviance: 723.4     AIC: 741.4
```

```
summary(logreg_full)
```

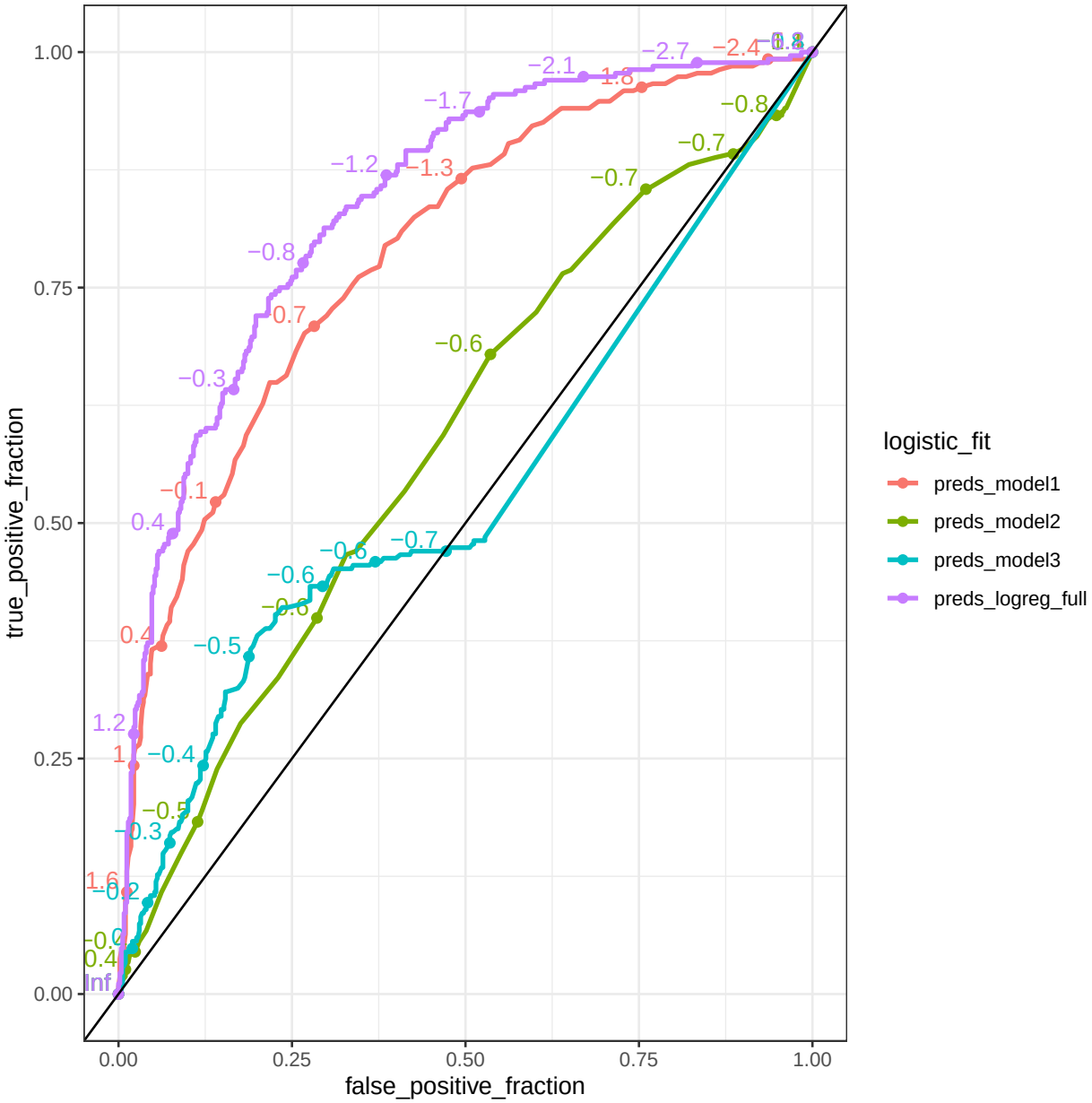
```
##
## Call:
## glm(formula = full_formula, family = "binomial", data = diabetes_dt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5566  -0.7274  -0.4159   0.7267   2.9297
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.4046964  0.7166359 -11.728  < 2e-16 ***
## Pregnancies               0.1231823  0.0320776   3.840 0.000123 ***
## Glucose                   0.0351637  0.0037087   9.481  < 2e-16 ***
## BloodPressure            -0.0132955  0.0052336  -2.540 0.011072 *
## SkinThickness             0.0006190  0.0068994   0.090 0.928515
## Insulin                  -0.0011917  0.0009012  -1.322 0.186065
## BMI                       0.0897010  0.0150876   5.945 2.76e-09 ***
## DiabetesPedigreeFunction  0.9451797  0.2991475   3.160 0.001580 **
## Age                       0.0148690  0.0093348   1.593 0.111192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 723.45  on 759  degrees of freedom
## AIC: 741.45
##
## Number of Fisher Scoring iterations: 5
```

```
diabetes_dt[, preds_logreg_full := predict(logreg_full)]
```

```
ggplot(diabetes_dt, aes(x=preds_logreg_full, fill=Outcome)) +
  geom_histogram(position="dodge", bins=50) + theme_bw()
```




```
plot_dt <- diabetes_dt[,.(Outcome, preds_model1, preds_model2, preds_model3, preds_logreg_full )] %>%
             melt(id.vars="Outcome", variable.name="logistic_fit", value.name="response")

# Plot roc curves for each model
ggroc <- ggplot(plot_dt, aes(d=as.numeric(Outcome), m=response, color=logistic_fit)) +
           geom_roc() +
           geom_abline() + theme_bw()
ggroc
```




```
# Overlapping histograms between positive and negative classes
# ROC curve is better for full model
```

Section 02 - Random Forests on Diabetes Dataset

1. Build a decision tree using the `rpart` function from the library `rpart` for predicting the `Outcome` given all feature variables. You can use the following command for this:

```
dt_classifier <- rpart(full_formula,
                       data =diabetes_dt,
                       control = rpart.control(minsplit = 3, cp = 0.001))
diabetes_dt
```

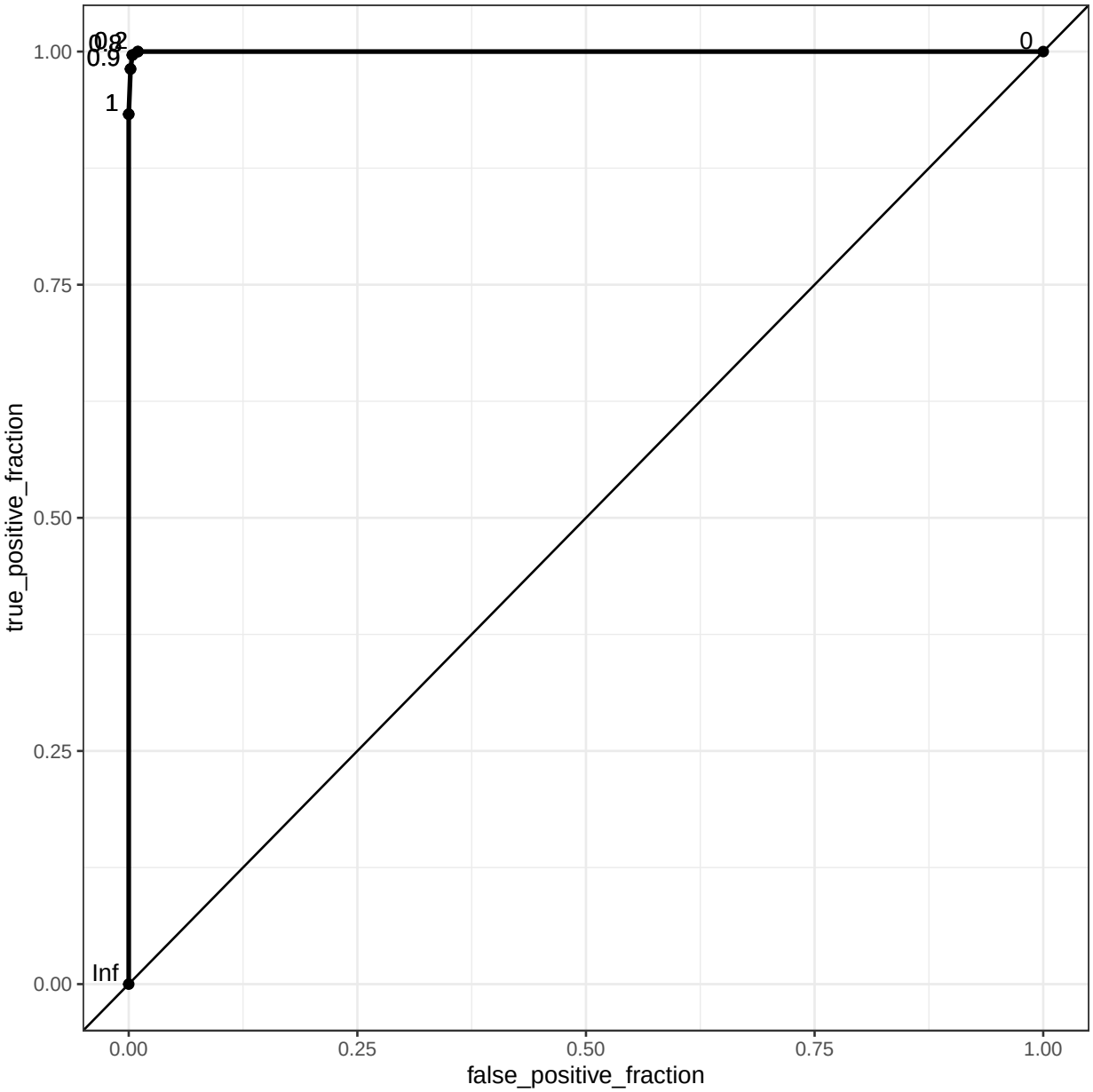
```
##      Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
```

```
##   1:           6     148            72            35       0 33.6
##   2:           1      85            66            29       0 26.6
##   3:           8     183            64             0       0 23.3
##   4:           1      89            66            23      94 28.1
##   5:           0     137            40            35     168 43.1
##  ---
## 764:          10     101            76            48     180 32.9
## 765:           2     122            70            27       0 36.8
## 766:           5     121            72            23     112 26.2
## 767:           1     126            60             0       0 30.1
## 768:           1      93            70            31       0 30.4
##      DiabetesPedigreeFunction Age Outcome preds_model1 preds_model2
##   1:                    0.627  50       1    0.2551290   -0.6055098
##   2:                    0.351  31       0   -2.1308723   -0.6500583
##   3:                    0.672  32       1    1.5806852   -0.6649078
##   4:                    0.167  21       0   -1.9793802   -0.6500583
##   5:                    2.288  33       1   -0.1614744   -0.8431018
##  ---
## 764:                    0.171  63       0   -1.5249037   -0.5758108
## 765:                    0.340  27       0   -0.7295700   -0.6203593
## 766:                    0.245  30       0   -0.7674430   -0.6055098
## 767:                    0.349  47       1   -0.5780778   -0.6946068
## 768:                    0.315  23       0   -1.8278880   -0.6203593
##      preds_model3 preds_logreg_full
##   1:   -0.8145101         0.9530421
##   2:   -0.8145101        -2.9734114
##   3:   -0.8145101         1.3658083
##   4:   -0.5984182        -3.1365417
##   5:   -0.4283033         2.2217292
##  ---
## 764:   -0.4007171        -0.7670604
## 765:   -0.8145101        -0.7585149
## 766:   -0.5570389        -1.5826843
## 767:   -0.8145101        -0.9199087
## 768:   -0.8145101        -2.5561608
```

Note that `cp` determines when the splitting up of the decision tree stops and `minsplit` determines the minimum amount of observations in a leaf of the tree.

2. Plot a ROC curve for the decision tree. What do you observe?

```
# Save predictions
diabetes_dt[, preds_dt := predict(dt_classifier, type="prob")[,2]]

# Plot roc curve for decision tree
ggroc <- ggplot(diabetes_dt, aes(d=as.numeric(Outcome), m=preds_dt)) +
            geom_roc() +
            geom_abline() + theme_bw()
ggroc
```

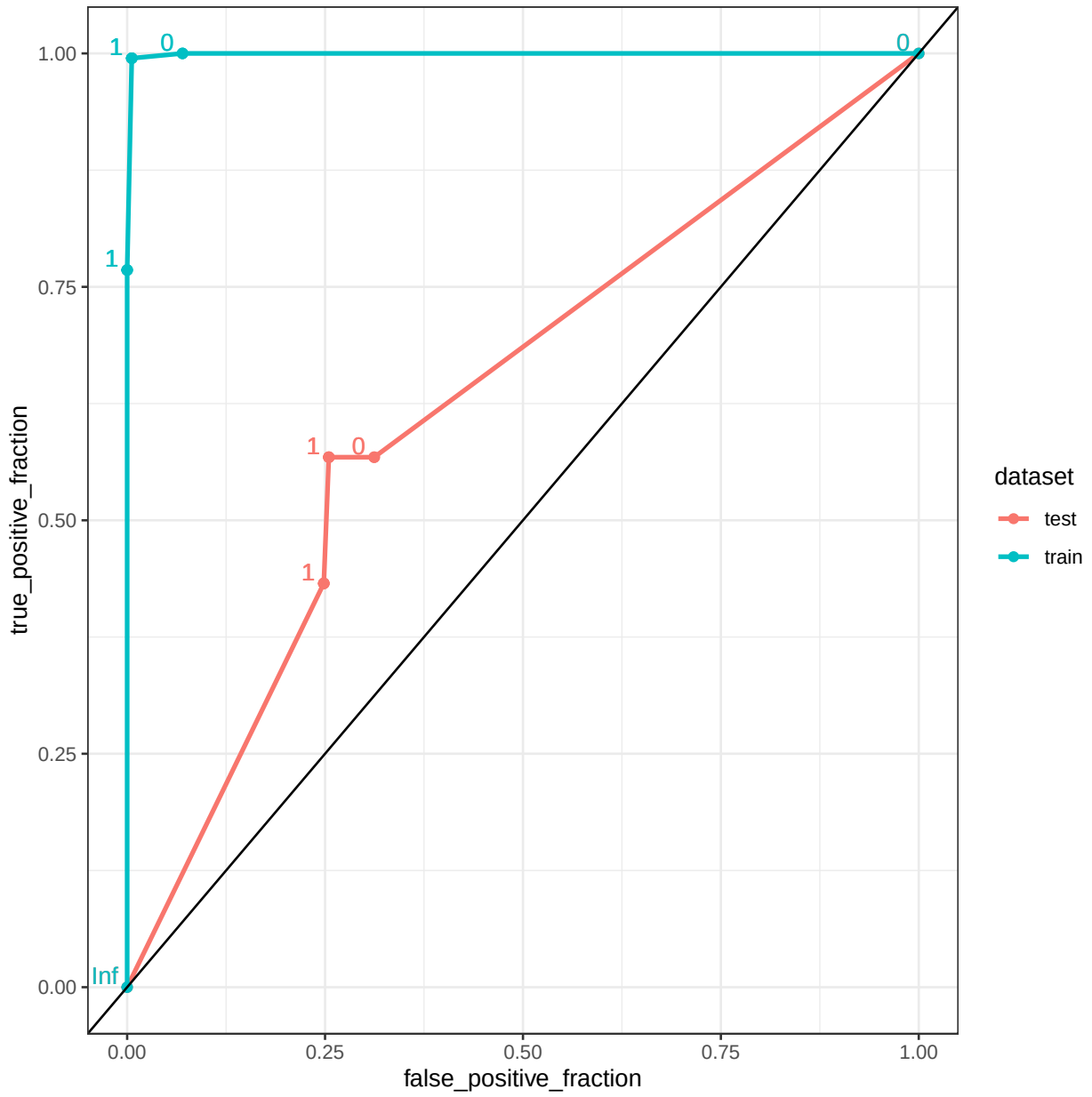



```
# The decision tree perfectly fits the dataset... is it too good to be true?
```

3. Build a second decision tree model this time using a train-test split strategy. This means that you will use 70% of the data for training and 30% of the data for testing. Plot the ROC curves for the performance on the training and on the test dataset. What do you conclude from this?

```
## 70% of the data for training
smp_size <- floor(0.70 * nrow(diabetes_dt))

## set the seed to make your partition reproducible
set.seed(13)
train_ind <- sample(seq_len(nrow(diabetes_dt)), size = smp_size)

# label train and test datasets
diabetes_dt[train_ind, dataset:="train"]
```

```r
diabetes_dt[-train_ind, dataset:="test"]


## train on training dataset
dt_classifier <- rpart(full_formula,
                       data =diabetes_dt[train_ind],
                       control = rpart.control(minsplit = 3, cp = 0))

# get predictions for both train and test set
diabetes_dt[, preds_dt := predict(dt_classifier, type="prob",
                                  # predict on all data, not only on trainset
                                  newdata=diabetes_dt)[,2]]

ggroc <- ggplot(diabetes_dt, aes(d=as.numeric(Outcome), m=preds_dt, color=dataset)) +
            geom_roc() +
            geom_abline() + theme_bw()
ggroc
```




```
# For plotting the decision tree
#library(partykit)
#as.party(dt_classifier)

#diabetes_dt[, unique(preds_dt)] --> cutoffs in ROC curve get rounded so be carefull with this.

# The model strongly overfits to the training dataset...
# we observe poor generalization to the test set...
```

4. In the lecture we learned that random forests are more robust to overfitting. Build a random forest using the `randomForest` function from the library `randomForest` for predicting the Outcome given all feature variables using the same train-test split strategy from before. Set the following values for the following hyper-parameters:

- `ntree = 200` for the number of trees in the forest,

- `nodesize = 20` for the maximum amount of leaf nodes,

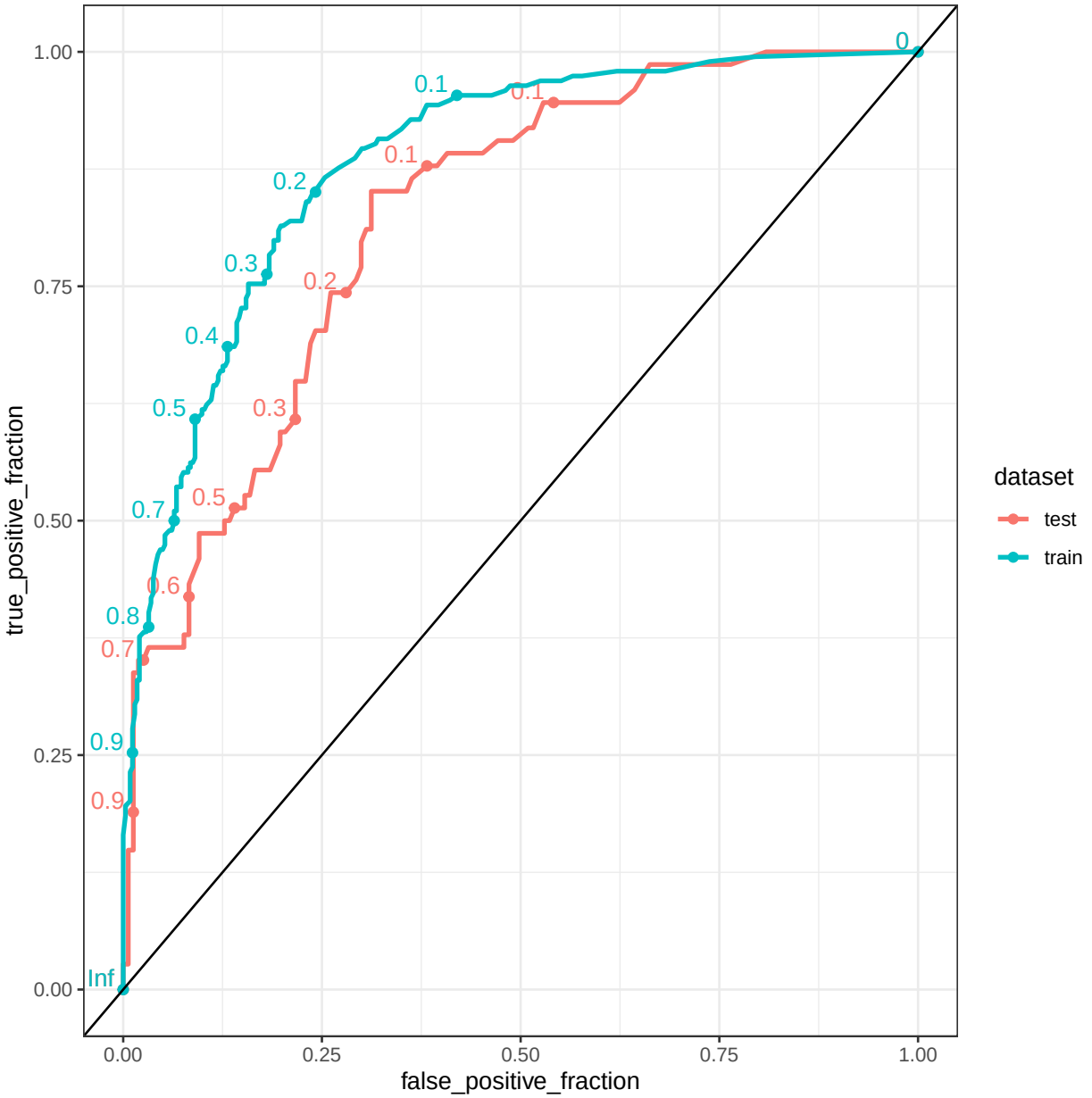
- `maxnodes = 7` for the minimum size of leaf nodes and
- `mtry = 5` for the number of variables randomly sampled as candidates at each split.

Plot the ROC curves for test and train set for the build random forest.

```
rf_classifier <- randomForest(## Define formula and data
                              full_formula,
                              ## Train only on trainset
                              data=diabetes_dt[train_ind],

                              ## Hyper parameters
                              ntree = 200,     # Define number of trees
                              nodesize = 20,  # Minimum size of leaf nodes
                              maxnodes = 7, # Maximum number of leaf nodes
                              mtry = 5, # Number of feature variables as candidates for each split
                              sampsize=length(train_ind),

                              ## Output the feature importances
                              importance=TRUE)

rf_classifier
```

```
##
## Call:
##  randomForest(formula = full_formula, data = diabetes_dt[train_ind],      ntree = 200, nodesize = 20
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 24.58%
## Confusion matrix:
##     0   1 class.error
## 0 299  44   0.1282799
## 1  88 106   0.4536082
```

```
diabetes_dt[, preds_rf := predict(rf_classifier, type="prob",
                                  ## Predict on all data
                                  newdata=diabetes_dt)[,2]]

# Plot roc curves for each model
ggroc <- ggplot(diabetes_dt, aes(d=as.numeric(Outcome), m=preds_rf, color=dataset)) +
           geom_roc() +
           geom_abline() + theme_bw()
ggroc
```




```
calc_auc(ggroc)
```

```
##   PANEL group       AUC
## 1     1     1 0.8210966
## 2     1     2 0.8825629
```

```
## Similar performance for both train and test set, no overfitting.
```

5. [OPTIONAL] Try changing the hyper-parameters with the aim of achieving a better performance on the test set evaluated with the same ROC curve as before.

```
# Hyper-parameter tuning is an important task in machine learning
# Usually default parameters work fine
# Otherwise, there are strategies such as (random) grid search
# for finding optimal hyper parameters (out of scope in this lecture)
```

Section 03 - Cross Validation on the Diabetes Dataset

1. Implement a 5-fold cross-validation on the diabetes dataset for building a logistic regression model using all feature variables. Obtain 5-fold cross-validated sensitivity, specificity and AUC using the `caret` package.

```
diabetes_dt[, Outcome:= ifelse(Outcome==1, "yes", "no")]
# somehow trainControl does not like factors...
# so we convert it to string

# generate control structure
fitControl <- trainControl(method = "cv",
                           number = 5, # number of fols
                           classProbs=TRUE, # display class probabilities
                           summaryFunction = twoClassSummary)
# run CV
logreg_cv <- train(full_formula,
                   data = diabetes_dt,
                  ## model specification
                  method = "glm", # we want a logistic regression
                  family = "binomial",
                  ## validation specification
                  trControl = fitControl,
                  ## Specify which metric to optimize
                  metric = "ROC")
logreg_cv
```

```
## Generalized Linear Model
##
## 768 samples
##   8 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 615, 614, 614, 614, 615
## Resampling results:
##
##   ROC        Sens   Spec
##   0.8279804  0.886  0.5709993
```

2. What is the fold with the highest AUC?

```
metrics_dt <- as.data.table(logreg_cv$resample)
metrics_dt[order(-ROC)]
```

```
##          ROC Sens      Spec Resample
## 1: 0.8575472 0.87 0.6226415    Fold1
## 2: 0.8337037 0.91 0.4814815    Fold4
## 3: 0.8271698 0.90 0.5471698    Fold5
## 4: 0.8240741 0.89 0.6111111    Fold3
## 5: 0.7974074 0.86 0.5925926    Fold2
```